A Function-wise Pre-training Technique for Constructing a Deep Neural Network based Spectral Model in Statistical Parametric Speech Synthesis

A FUNCTION-WISE PRE-TRAINING TECHNIQUE FOR CONSTRUCTING A DEEP NEURAL NETWORK BASED SPECTRAL MODEL IN STATISTICAL PARAMETRIC SPEECH SYNTHESIS

Shinji Takaki, Junichi Yamagishi

National Institute of Informatics
Japan

Zhenzhou Wu[†]

Institute of High Performance Computing
A-STAR, Singapore

ABSTRACT

This paper presents a technique for spectral modeling using deep neural networks (DNNs) for statistical parametric speech synthesis. In statistical parametric speech synthesis systems, spectra are generally represented by low-dimensional spectral envelope parameters such as cepstra and line spectral pairs (LSPs), and the parameters are statistically modeled using hidden Markov models (HMMs) or DNNs. In this paper, we propose a statistical parametric speech synthesis system that directly models high-dimensional spectral amplitudes by using the DNN framework to improve the modeling of spectral fine structures. We combine two DNNs, i.e., the first for data-driven feature extraction from spectral amplitudes pre-trained using an auto-encoder and the second for acoustic modeling into a large network and optimize the networks together to construct a single DNN that directly synthesizes spectral amplitude information from linguistic features. Experimental results showed that the proposed technique increased the quality of synthetic speech.

Index Terms— Speech synthesis, DNN, Auto-encoder, Spectral amplitude

1. INTRODUCTION

Current statistical parametric speech synthesis typically uses hidden Markov models (HMMs) to represent the probability densities of speech trajectories given text [1]. This is a well-established method and this framework can directly be applied to new languages. It also offers interesting advantages in terms of flexibility and a compact footprint [2, 3, 4, 5]. It is well known, however, that speech synthesized from statistical models still sounds somehow artificial and less natural than speech synthesized with the best unit-selection systems.

Recently, research on statistical speech synthesis has been significantly advanced due to deep neural networks (DNNs) with many hidden layers. For instance, DNNs have been applied to acoustic modeling. Zen et al. [6] use DNN to learn the relationship between input texts and extracted features instead of decision tree-based state tying. Restricted Boltzmann machines or deep belief networks have been used to model the output probabilities of hidden Markov model (HMM) states instead of Gaussian mixture models (GMMs) [7]. Recurrent neural networks and long-short term memories have been used for prosody modeling [8] and acoustic trajectory modeling [9]. In addition, an auto-encoder neural network has also been used to extract low dimensional excitation parameters [10].

, and the averaging effects of statistical models have often been said to remove the spectral fine structures of natural speech. To improve the quality of synthetic speech, a stochastic postfilter approach has been proposed [11] where a DNN is used to model the conditional probability of the spectral differences between natural and synthetic speech. The approach was found to be able to reconstruct the spectral fine structures lost during modeling and it significantly improved the quality of synthetic speech [11]. In this experiment, the acoustic model was trained using lower dimensional spectral envelope features, while the DNN-based postfiler was trained using the spectral amplitudes obtained using the STRAIGHT vocoder [12]. From the experimental findings, we can hypothesize that the current statistical parametric speech synthesis may suffer from quality loss due to not only statistical averaging but also acoustic modeling using lower dimensional acoustic features. Lower dimensional acoustic features are also normally extracted in a speaker-independent deterministic fashion.

On the basis of this hypothesis, in this paper we present a new technique for constructing a DNN that directly synthesizes spectral amplitudes from linguistic features without using spectral parameters such as the mel-cepstrum. It is well known that there are many problems in training a DNN such as the local optima, vanishing gradients and so on [13]. However, it has been reported in the field of automatic speech recognition (ASR) that DNNs that deal with high-dimensional features, e.g., the FFT frequency spectrum, can be appropriately constructed using an efficient training

*This work was supported in part by EPSRC through Programme Grant EP/I031022/1 (NST) and EP/J002526/1 (CAF). Shinji Takaki was supported in part by NAVER Labs.

[†]Zhenzhou Wu performed this work while at the National Institute of Informatics, Japan.

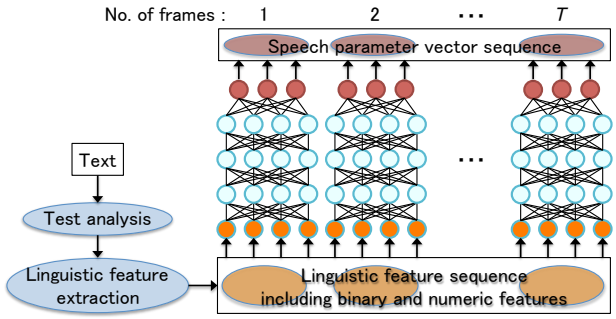


Fig. 1. A framework of DNN-based acoustic model.

technique such as pre-training [14].

Thus, in this paper we propose an efficient training technique to construct a DNN that directly synthesizes high-dimensional spectral amplitudes from input texts. The key idea is to stack two DNNs, an auto-encoder neural network for data-driven non-linear feature extraction from the spectral amplitudes and another network for acoustic modeling and context clustering. The proposed technique is regarded as a function-wise pre-training technique to construct the DNN-based speech synthesis system.

The rest of this paper is organized as follows. Section 2 reviews a DNN-based acoustic model for the statistical parametric speech synthesis. Section 3 describes a DNN-based acoustic feature extractor and spectrum re-generator. Section 4 explains the proposed technique of constructing a DNN that directly synthesizes the spectral amplitudes. The experimental conditions and results obtained from analysis-re-synthesis and text-to-speech synthesis experiments are presented in Sections 5, 6 and 7. Concluding remarks and future work are presented in Section 8.

2. DNN-BASED ACOUSTIC MODEL FOR STATISTICAL PARAMETRIC SPEECH SYNTHESIS

It is believed that the human speech production system has layered hierarchical structures to convert linguistic information into speech. To approximate such a complicated process, DNN-based acoustic models that represent the relationship between linguistic and speech features have been proposed for statistical parametric speech synthesis [6, 7, 8, 9]. This section briefly reviews one of the state-of-the-art DNN-based acoustic models [6].

Figure 1 illustrates the framework of the DNN-based acoustic model, where linguistic features obtained from a given text are mapped to speech parameters by a DNN. The input linguistic features include binary answers to questions about linguistic contexts and numeric values, e.g., the number of words in a current phrase, the position of a current syllable in a word, and the duration of a current phoneme. In [6], the output speech parameters include spectral and excitation parameters and their time derivatives (dynamic features). By using pairs of input and output features obtained from training data, the parameters of the DNN can be trained with a stochastic gradient descend (SGD) [15]. Speech parameters can be predicted for an arbitrary text by utilizing a trained DNN using forward propagation.

3. DNN-BASED ACOUSTIC FEATURE EXTRACTION

This section describes a DNN-based acoustic feature extraction. An auto-encoder allows us to automatically extract robust low-dimensional features from high-dimensional spectral features in a non-linear, data-driven and unsupervised way.

3.1. Basic Auto-encoder

An auto-encoder is an artificial neural network that is used generally for learning a compressed and distributed representation of a dataset. It consists of an encoder and a decoder. The encoder in the basic one-hidden-layer auto-encoder maps an input vector $\mathbf{x}$ to a compressed hidden representation $\mathbf{y}$ as follows:

$$\mathbf{y} = f_\theta(\mathbf{x}) = s(\mathbf{W}\mathbf{x} + \mathbf{b}), \tag{1}$$

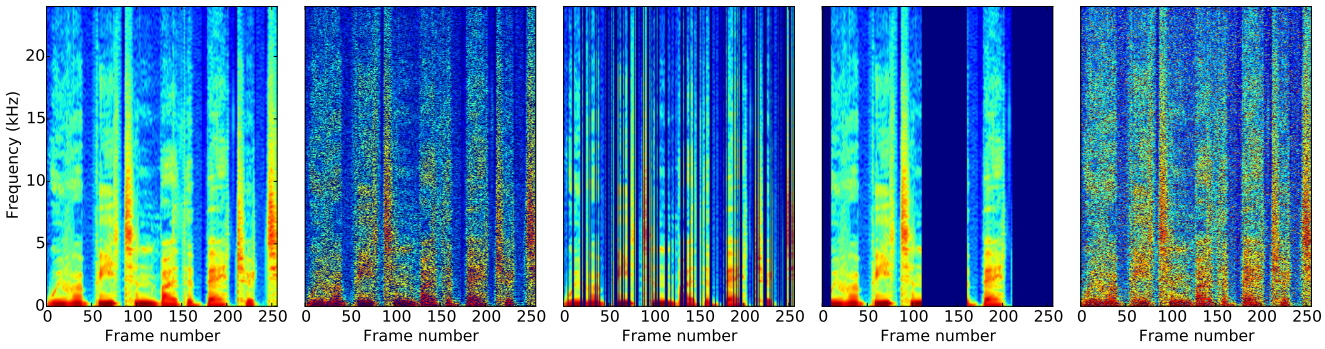
where $\theta = \{\mathbf{W}, \mathbf{b}\}$. $\mathbf{W}$ is a $m \times n$ weight matrix and $\mathbf{b}$ is a m dimension bias vector. The function s is a non-linear transformation on the linear mapping $\mathbf{W}\mathbf{x} + \mathbf{b}$. We typically use sigmoid, tanh or ReLU for the non-linear transformation. The output from the encoder is a low-dimensional representation $\mathbf{y}$, which is then passed into the decoder $g_{\theta'}$ to reconstruct back to the original dimension. The reconstruction is performed by a linear mapping followed by an arbitrary linear or non-linear function t that employs an $n \times m$ weight matrix $\mathbf{W}'$ and a bias vector of dimensionality n as follows:

$$\mathbf{z} = g_{\theta'}(\mathbf{y}) = t(\mathbf{W}'\mathbf{y} + \mathbf{b}'), \tag{2}$$

where $\theta' = \{\mathbf{W}', \mathbf{b}'\}$. The parameters $\{\theta, \theta'\}$ are optimized such that the reconstructed $\mathbf{z}$ is as close as possible to the original $\mathbf{x}$. mean squared error (MSE) is typically used as the objective function for SGD to measure the distance between the input vector $\mathbf{x}$ and the reconstructed vector $\mathbf{z}$.

3.2. Denoising Auto-encoder

The denoising auto-encoder is a variant of a basic auto-encoder. It has been reported that a denoising auto-encoder can extract features more robustly than a basic auto-encoder [16]. In the denoising auto-encoder, the original data $\mathbf{x}$ are first corrupted to $\tilde{\mathbf{x}}$ before they are mapped to a low-dimensional hidden representation $f_\theta(\tilde{\mathbf{x}})$ by an encoder. The decoder then reconstructs the low-dimensional hidden representation into the original dimension $\mathbf{z}$. The denoising auto-encoder is trained such that the reconstructed $\mathbf{z}$ is as



(a) Original (b) Mask-out (c) Black-out (d) Batch-out (e) Gaussian

Fig. 2. These figures shows parts of original and noise added spectrograms. Dark blue points in this figure indicate masked regions.

close as possible to the original data $\mathbf{x}$. Note that it is only during training that the denoising auto-encoder is used to reconstruct the original $\mathbf{x}$ from the corrupted $\tilde{\mathbf{x}}$.

3.2.1. Noise

When applying noise $\boldsymbol{\epsilon}$ to the dataset $\mathbf{x}$ for training a denoising auto-encoder, the noise can either be multiplicative

$$\tilde{\mathbf{x}} = \mathbf{x} \odot \boldsymbol{\epsilon}, \tag{3}$$

where $\odot$ represents element-wise multiplication, or additive. Adding noise during training was shown to regularize weight, which prevented saturation of non-linear functions that had a vanishing gradient at large inputs such as the sigmoid or tanh function [17]. Preventing saturation of the non-linear transformation allowed errors to effectively back-propagate to the lower layers. Also, Gaussian and mask-out noise was shown by [16] to yield more robust bottleneck features for the auto-encoder. In this paper, we used two common noise Gaussian and mask-out, and two proposed noise which we termed black-out and batch-out for training the denoising auto-encoder. We confirm that adding noises improve the robustness for the bottleneck feature and reduce the reconstruction error.

3.2.2. Mask-Out Noise

Mask-out noise is independent element-wise multiplicative noise as in Eq. 3, where $\boldsymbol{\epsilon}$ follows the Bernoulli distribution of probability p.

$$\boldsymbol{\epsilon} \sim Bernoulli(p). \tag{4}$$

3.2.3. Gaussian Noise

Gaussian noise is additive noise that corrupts each value of the input vector into a neighbouring value following a Gaussian distribution.

$$\boldsymbol{\epsilon} \sim N(\mu, \sigma^2). \tag{5}$$

3.2.4. Black-Out Noise

Black-out noise is similar to mask-out noise, and it randomly sets the values of an entire example to zero. Black-out noise is defined as:

$$\tilde{\mathbf{x}} = \mathbf{x} \times \epsilon, \tag{6}$$

where the scalar ϵ follows the Bernoulli distribution with probability p.

$$\epsilon \sim Bernoulli(p). \tag{7}$$

3.2.5. Batch-Out Noise

For SGD training following [15], we often use mini-batches with size 50 to 200 examples for forward propagation through the network before one error back-propagation. Batch-out noise is similar to black-out noise except that batch-out noise sets the values of an entire mini-batch $\mathbf{x}_{bat}$, i.e., sequential examples, to zero:

$$\tilde{\mathbf{x}}_{bat} = \mathbf{x}_{bat} \times \epsilon, \tag{8}$$

where scalar ϵ also follows the Bernoulli distribution with probability p. We can think of black-out and batch-out noise as additive noise in the time dimension. Figure 2 shows examples of original and noise added spectrograms. In this figure, the dark blue points indicate regions with zero values.

3.3. Deep Auto-encoder

An auto-encoder can be made deeper by stacking multiple layers of encoders and decoders to form a deep architecture. Pre-training is widely used for constructing a deep auto-encoder. In pre-training, the number of layers in a deep auto-encoder increases twice as compare to a deep neural

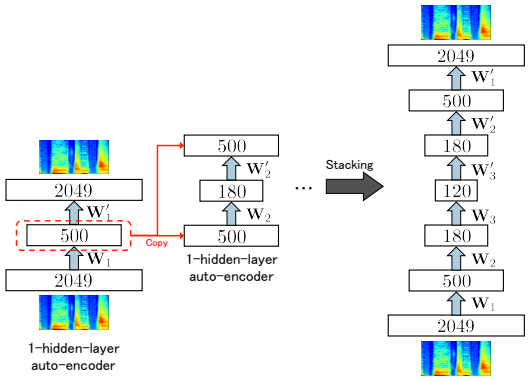


Fig. 3. Greedy layer-wise pre-training for constructing a deep auto-encoder.

network (DNN) when stacking each pre-trained unit. It has been reported that fine-tuning with back-propagation through a deep auto-encoder is ineffective due to vanishing gradients at the lower layers [13]. To overcome this issue, we restrict the decoding weight as the transpose of the encoding weight following [15], that is, $\mathbf{W}' = \mathbf{W}^T$ where $\mathbf{W}^T$ denotes the transpose of $\mathbf{W}$. Each layer of a deep auto-encoder can be pre-trained greedily to locally minimize the reconstruction loss of the data. Figure 3 shows the procedure for constructing a deep auto-encoder using layer-by-layer pre-training. In pre-training, a one-hidden-layer basic auto-encoder is trained and the encoding output of the locally trained layer is used as the input to the next basic auto-encoder with a smaller bottleneck layer. After all layers are pre-trained, they are stacked and fine-tuned with SGD to minimize the reconstruction error over the entire dataset. Note that mean squared error (MSE) is used as the loss function for both pre-training and fine-tuning.

Each layer of a deep auto-encoder can be greedily pre-trained to locally minimize the reconstruction loss $L(\mathbf{x}, \mathbf{z})$ of the data. During pre-training, we first train each one-hidden-layer basic auto-encoder, and then the encoding output of the locally trained layer is used as the input for the next layer. This layer-wise training is repeated until the required layer size is obtained. The encoding, decoding and loss functions of each layer are represented as follows:

$$\begin{aligned}
&\texttt{Layer 1:}\\
&\mathbf{y}_1 = f_{\mathbf{W}_1,\mathbf{b}_1}(\mathbf{x}),\\
&\mathbf{z}_1 = g_{\mathbf{W}_1',\mathbf{b}_1'}(\mathbf{y}_1),\\
&L(\mathbf{x}, \mathbf{z}_1) = |\mathbf{x} - \mathbf{z}_1|^2,\\
&\texttt{Layer k (k>1):}\\
&\mathbf{y}_k = f_{\mathbf{W}_k,\mathbf{b}_k}(\mathbf{y}_{k-1}),\\
&\mathbf{z}_k = g_{\mathbf{W}_k',\mathbf{b}_k'}(\mathbf{y}_k),\\
&L(\mathbf{y}_{k-1}, \mathbf{z}_k) = |\mathbf{y}_{k-1} - \mathbf{z}_k|^2.
\end{aligned} \tag{9}$$

Note that during the pre-training of the deep denoising auto-encoder, the input $\mathbf{x}$ and $\mathbf{y}_k$ for each layer are corrupted to $\tilde{\mathbf{x}}$ and $\tilde{\mathbf{y}_k}$ respectively. After all layers are pre-trained, all the pre-trained layers are stacked for constructing a deep denoising auto-encoder in the same way as the deep auto-encoder.

The purpose of fine-tuning is to minimize the reconstruction error $L(\mathbf{x}, \mathbf{z})$ over the entire dataset and a model architecture using error backpropagation [18]. We use the mean square error (MSE) for the loss function of a deep auto-encoder and it is represented as follows:

$$E = \sum_{i=1}^{N} |\mathbf{x}^{(i)} - \mathbf{z}^{(i)}|^2, \tag{10}$$

where N is the total number of training examples. The partial derivatives w.r.t weight $w_{i,j}^{(l)}$ is represented as follows:

$$\frac{\partial E}{\partial w_{i,j}^{(l)}} = \frac{\partial E}{\partial t_j^{(l)}} \frac{\partial t_j^{(l)}}{\partial w_{i,j}^{(l)}}, \tag{11}$$

where $t_j^{(l)}$ is the fan-in input to neuron j in layer l, and $\frac{\partial t_j^{(l)}}{\partial w_{i,j}^{(l)}} = o_i^{(l-1)}$, where $o_i^{(l-1)}$ is the output from neuron i at layer $l-1$. $\frac{\partial E}{\partial t_j^{(l)}}$ is the error transfer function that can be calculated following

$$\frac{\partial E}{\partial t_j^{(l)}} = \frac{\partial o_i^{(l)}}{\partial t_j^{(l)}} \sum_{j=1}^{J} \frac{\partial E}{\partial t_j^{(l+1)}} \frac{\partial t_j^{(l+1)}}{\partial o_i^{(l)}}, \tag{12}$$

where $\frac{\partial t_j^{(l+1)}}{\partial o_i^{(l)}} = w_{i,j}^{(l)}$. For the tanh function we have $\frac{\partial o_i^{(l)}}{\partial t_j^{(l)}} = \text{sech}^2(t_i^{(l)})$. Once we have the gradients of the error function w.r.t to the weight parameters, we can fine-tune the network with error back propagation

3.4. Related work using auto-encoder in the speech information processing

Deep auto-encoder based bottleneck features have been used by several groups for ASR [19, 20] and a deep denoising auto-encoder has also verified for noise-robust ASR [21] or reverberant ASR tasks [22, 23]. Techniques that are closely related to this paper are a spectral binary coding approach using a deep auto-encoder proposed by Deng et al. [24] and a speech enhancement approach using a deep denoising auto-encoder where Lu et al. tried to reconstruct a clean spectrum from a noisy spectrum [25]. The approach proposed here is also related to heteroscedastic linear discriminant analysis (HLDA) [26, 27] and probabilistic linear discriminant analysis (PLDA) [28, 29, 30].

In the field of speech synthesis, similar auto-encoder based bottleneck features were tested for excitation parameters [10, 31] and a ClusterGen speech synthesizer [32]. Our idea is different from theirs and we stack the decoder part of the deep auto-encoder onto another DNN for acoustic modeling, as described in the next section.

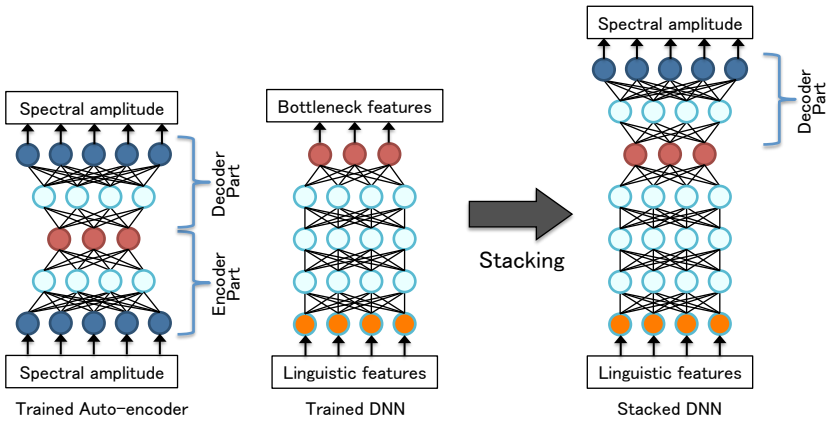


Fig. 4. Procedure for constructing a DNN-based spectral model based on a deep auto-encoder and a DNN-based acoustic model.

4. PROPOSED DNN-BASED SPECTRAL MODELING

The DNN-based acoustic model described in Section 2 may be used for the direct spectral modeling by substituting the output of the network from the mel-cepstrum to the spectrum. However, the dimension of the spectrum is much higher than that of the mel-cepstrum. For a speech signal at 48 kHz, the order of the mel-cepstral analysis that is typically used is around 60-dim, whereas the dimension of the FFT spectrum is 2049. Because of this high dimensional data, a more efficient training technique is needed to construct a DNN that directly represents the relationship between linguistic features and spectra. In this paper, we therefore propose a function-wise pre-training technique where we explicitly divide the general flow of the statistical parametric speech synthesis system into a few sub-processes, construct and optimize a DNN for each task individually, and stack the individual networks for the final optimization.

Figure 4 shows the procedure for constructing the proposed DNN-based spectral model. The details on the three steps in the proposed technique are below:

Step 1. Train a deep auto-encoder using spectral amplitudes and extract bottleneck features for the DNN-based acoustic model to be used in Step 2. Layer-wise pre-training or other initialization may be used for the learning of the deep auto-encoder.

Step 2. Train a DNN-based acoustic model using the bottleneck features extracted in Step 1. Layer-wise pre-training or other initialization may be used for learning the DNN.

Step 3. Stack the trained DNN-based acoustic model for bottleneck features and the decoder part of the trained deep auto-encoder as shown in Figure 4 and optimize the whole network.

A DNN that represents the relationship between linguistic features and spectra is constructed based on a DNN-based spectral generator and a DNN-based acoustic model using the bottleneck features. After this proposed pre-training, we fine-tune the DNN to minimize error over the entire dataset using pairs of linguistic features and spectral amplitudes in training data with SGD.

5. EXPERIMENT I: ANALYSIS-RE-SYNTHESIS

Although our key idea is to stack the auto-encoder onto another DNN for acoustic modeling as mentioned earlier, it is also important to perceptually evaluate the auto-encoder solely as a data-driven feature extractor from the spectral amplitudes. We therefore have first conducted analysis-re-synthesis (copy-synthesis) experiments and compared the auto-encoder based extraction approach with the conventional mel-cepstral analysis which is based on a linear discrete cosine transform of the spectrum.

5.1. Experimental configurations

The dataset we used consists of 4,558 short audio waveforms uttered by a female professional who is a native-English speaker and each waveform is around five seconds long. All data were sampled at 48 kHz. In analysis-re-synthesis experiment, we divided the database into three subsets, that is, training (3,676 utterances), validation (441 utterances) and test (441 utterances). The training subset was used as training data to build the auto-encoder, the validation subset was used as a stopping criterion during training to prevent over-fitting, and the test subset was used for measuring log-spectral distortion and a listening test.

For each waveform, we first extracted its frequency spectra using the STRAIGHT vocoder with 4096 FFT points. We then extracted the low dimensional feature from each 2049-dim STRAIGHT spectrum using the deep auto-encoder. We

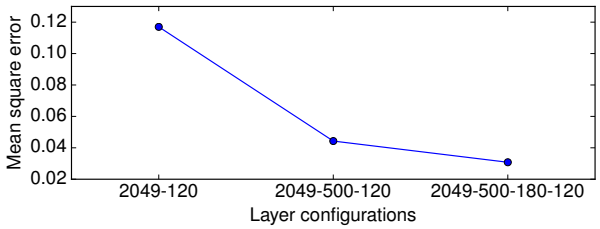


Fig. 5. Reconstruction mean square errors for auto-encoders with different architectures but same bottleneck dimension. Layer configurations represent the number of units in the decoder part of the auto-encoder.

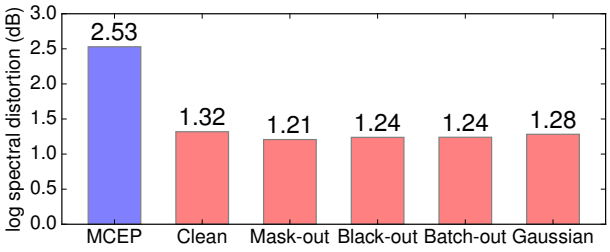


Fig. 6. Log spectral distortion between the original spectral amplitudes and spectral amplitudes re-synthesized using the mel-cepstral analysis and deep auto-encoders with and without a denoising process.

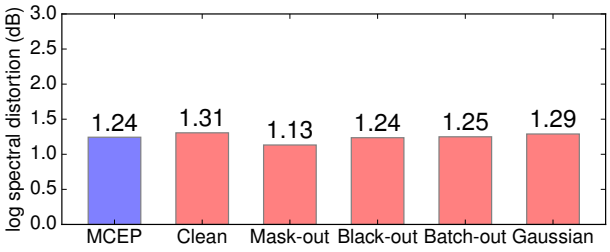


Fig. 7. Log spectral distortion between the original frequency-warped spectral amplitudes and frequency-warped spectral amplitudes re-synthesized using the mel-cepstral analysis and deep auto-encoders with and without a denoising process.

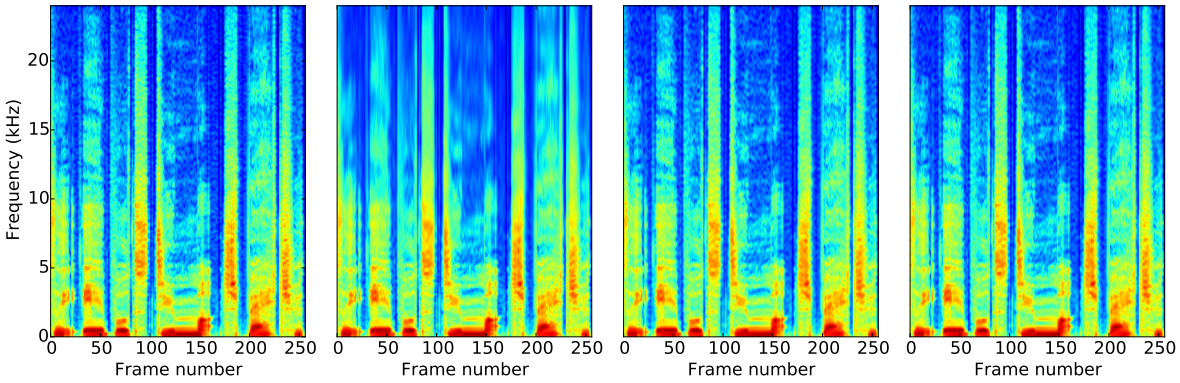
used log frequency-warped spectral amplitudes that were pre-processed with global contrast normalization (GCN) as inputs to the deep auto-encoder. The objective of GCN was to normalize the length of each example vector to one, so that after GCN, all the example vectors that were a scalar factor of each other were mapped to the same unit-length vector. We used the tanh function for all units of hidden and output layers of the deep auto-encoder. This was because we down-scaled all the input values to the range of [-1,1], which fell into the range for the tanh function after applying GCN on the log frequency-warped spectral amplitudes. For comparison of the proposed method, we extracted frequency-warped cepstral coefficients that used the same dimensions as that of the deep auto-encoder. The Bark scale was used for frequency warping. All other acoustic features such as log F0 and 25-dim aperiodicity band energies were the same for all the systems. We synthesized speech samples from spectrum amplitudes, F0 features and aperiodicity energies using the STRAIGHT vocoder. Thus cepstral vectors were converted into spectrum amplitudes to use the STRAIGHT vocoder in the synthesis phase.

Preference tests were conducted in subjective experiments. Nineteen subjects participated in these listening tests. Fifteen sentences were randomly selected from the test set for each subject. The subjects were native speakers of English and they were asked to listen to a pair of speech samples to answer which samples sound more natural. The experiments were carried out using headphones in quiet rooms.

5.2. Experimental results

Figure 5 plots the reconstruction mean square errors at each hidden layer of the deep auto-encoder calculated with the test set. It shows that errors decrease with more hidden layers, and that the deep auto-encoder is better than the shallow auto-encoder with the same bottleneck dimensions. Based on these results, we used 2049-500-180-120 as the deep auto-encoder architecture to produce the 120-dim acoustic features for measuring log-spectral distortion and listening test. The hyper-parameters used for the layer-by-layer pre-training were randomly searched using the validation set and the set of values that produced the best results were selected. Table 1 shows the hyper-parameters for the auto-encoders used in the analysis-re-synthesis experiments. For comparison, 119 dimensional frequency-warped cepstrum (plus 0th) were also extractied.

Figures 6 and 7 show the log spectral distortion between the original spectral amplitudes and spectral amplitudes re-synthesized using the mel-cepstral analysis and the deep auto-encoders with and without a denoising process calculated on the test subset in the linear frequency domain and warped frequency domain, respectively. In these figure, the *MCEP* refers to the mel-cepstral analysis, *Clean* refers to the spectral distortion of the reconstructed spectrum from the original spectrum produced by the deep auto-encoder trained on the original dataset. *Mask-out*, *Black-out*, *Batch-out* and *Gaussian* refer to the same distortion error for the reconstructed spectrum but using deep denoising auto-encoders trained on each respective noise. We can observe from Figure 6 that the deep auto-encoder has reduced distortion significantly compared to the mel-cepstral analysis and the denoising version reduced distortion even further than the clean version. We can see from Figure 7 that the deep denoising auto-encoder



(a) Original (b) MCEP (c) Clean (d) Mask-out

Fig. 8. Original and re-synthesized spectrograms using each technique.

Table 1. Table lists hyper-parameters used for training each model in analysis-re-synthesis experiments. lr: learning rate, m: momentum, b: batch size, s: numpy random variable weight initialization seed [33], and d: masking probability for each input dimension [16].

	layer dim	lr	m	b	s	d
Deep auto-encoder	2049-500	0.001	0.9	200	8963	N.A
	500-180	0.01	0.5	50	1902	N.A
	180-120	0.01	0.9	50	6555	N.A
	Finetune	0.01	0.5	150	9781	N.A
Deep de-noising auto-encoder	2049-500	0.01	0.1	150	5252	0.1
	500-180	0.01	0.5	150	7514	0.1
	180-120	0.01	0.9	100	594	0.5
	Finetune	0.001	0.9	100	2208	N.A

using mask noise has also reduced distortion more in the frequency-warped domain than that with the mel-cepstral analysis. Although the marginal gain by the denoising process may not be surprising, it is interesting to investigate why the deep auto-encoder has significantly less distortion than the mel-cepstrum and how the fine structure of the spectrum has recovered. We therefore looked into the spectrograms using re-synthesized spectral amplitudes using each technique, i.e., the mel-cepstral analysis (*MCEP*), the deep auto-encoder (*Clean*), and the deep denoising auto-encoder (*Mask-out*), which are shown in Figure 8. We can clearly see from these spectrograms that the deep auto-encoders could reconstruct high-frequency parts more precisely than those of the mel-cepstral analysis, which explains the lower spectral distortion.

We further conducted subjective tests to perceptually evaluate the auto-encoder. More precisely, we conducted two preference tests. In the first preference test, participants were asked to listen to a pair of speech samples generated

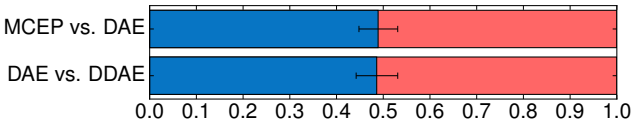


Fig. 9. Results from preference tests using analysis-re-synthesis speech samples. In this figure, MCEP, DAE and DDAE refer to the mel-cepstrum analysis, the deep auto-encoder and the deep denoising auto-encoder using mask-out noise respectively.

using the deep auto-encoder (DAE) or the mel-cepstral analysis (MCEP). In the second preference test, they were asked to compare the deep auto-encoder with the deep denoising auto-encoder using mask-out noise (DDAE).

Figure 9 shows the results for the two subjective preference tests with 95% confidence intervals. From the figure, perceptual difference between the mel-ceptrum analysis based speech samples and deep auto-encoder based speech samples was not statistically significant. This means that the bottleneck features found in a non-linear, data-driven and unsupervised way is as perceptually meaningful as mel-cepstral coefficients and this indicate that we can use the bottleneck features for acoustic modeling. The difference between clean and denoising auto-encoder was not statistically significant.

6. EXPERIMENT II: TEXT-TO-SPEECH SYNTHESIS

6.1. Experimental configurations

Again, our goal is to model high-dimensional spectral amplitudes in speech synthesis. However, in the second experiment, we use the low-dimensional spectral parameters extracted from the deep auto-encoder in conventional HMM and DNN-based speech synthesis systems, and we show that the

low-dimensional spectral parameters are suitable for statistical modeling.

In text-to-speech synthesis experiment, we used all the 4,558 utterances to train HMMs or DNNs. We used 180 sentences from a different dataset for the evaluation. In this experiment, we trained a symmetric five-hidden-layer auto-encoder without denoising processing where the numbers of units were 2049, 500, 60, 500 and 2049 in the hidden layers, considering the fact that the order of the mel-cepstral analysis typically used in statistical parametric speech synthesis is around 60-dim. We have used a hidden semi-Markov model for the HMM-based speech synthesis. The feature vectors for HMMs were comprised of 258 dimensions: 59 dimensional frequency-warped cepstrum (plus 0th) or 60 dimensional spectral parameters extracted from a deep auto-encoder, log f0, 25-dim aperiodicity band energies, and their dynamic and acceleration coefficients. For the DNN-based speech synthesis, continuous log f0 interpolated linearly for unvoiced regions and voiced/unvoiced parameters were used for excitation parameters. Thus, 259 dimensional features were used as output features of the DNN. The context-dependent labels were built using the pronunciation lexicon Combilex [34]. The linguistic features for DNN acoustic models were comprised of 897 dimensions: 858 dimensional binary features for categorical linguistic contexts, 36 numerical features for numerical linguistic contexts, and three numerical features for the position of the current frame and duration of the current phoneme. Phoneme boundaries were estimated with the HMM-based speech synthesis system and fixed in training DNNs for acoustic models. The linguistic features and spectral amplitudes in the training data were normalized for training DNNs. The input linguistic features were normalized to have zero-mean unit-variance, whereas the output spectral amplitudes were normalized to be within 0.0–1.0. We synthesized speech samples from spectrum amplitudes, F0 features and aperiodicity energies using the STRAIGHT vocoder. Thus synthesized cepstral vectors were converted into spectrum amplitudes to use the STRAIGHT vocoder. We used the sigmoid function for all the units of hidden and output layers of all DNNs.

Preference tests were conducted in subjective experiments. Twenty native speaking subjects and seven non-native speaking subjects participated in these listening tests. Fifteen and thirty sentences were randomly selected from the test set for each subject in native and non-native listening tests, respectively. The subjects were asked to listen to a pair of speech samples to answer which samples sound more natural. The experiments were carried out using headphones in a quiet room.

6.2. Experimental Results

Figures 10 and 11 show the subjective preference scores for both native and non-native listeners respectively where we have compared the proposed auto-encoder feature with the conventional mel-cepstral feature in each of the HMM-based speech synthesis and the DNN-based speech synthesis systems. We can see from the figures that synthetic speech using the proposed feature sounded more natural than that using the conventional mel-cepstral features in the DNN synthesis method for both the native and non-native listeners. Interestingly, non-native listeners have rated synthetic speech using the proposed feature slightly better than that using the conventional mel-cepstral features even in the HMM synthesis method.

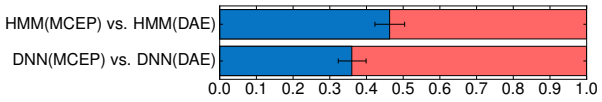


Fig. 10. Results from preference tests using text-to-speech samples (native listeners).

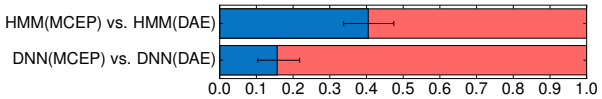


Fig. 11. Results from preference tests using text-to-speech samples (non-native listeners).

7. EXPERIMENT III: PROPOSED TEXT-TO-SPEECH SYNTHESIS

7.1. Experimental configurations

Finally, we evaluated the proposed stacking technique for spectral amplitude modeling in text-to-speech synthesis. We used 4,558 utterances and 180 sentences for training and evaluation in the same way as the previous text-to-speech experiment.

We compared three techniques; *CEPSTRUM* is the DNN that synthesizes cepstrum vectors, *SPECTRUM* have the same network structure as that of *CEPSTRUM*, but it directly outputs the spectral amplitudes, and *STACK* is the proposed DNN that synthesize spectrum amplitudes with the proposed pre-training framework. Figure 12 shows the structures of constructed DNNs for each technique. We trained five-hidden-layer DNN-based acoustic models for each technique. The number of units in each of the hidden layers was set to 1024. Random initialization was used in a way similar to [6]. We trained the symmetric five-hidden-layer auto-encoder without denoising processing for *STACK*. The numbers of units were 2049, 500, 60, 500 and 2049 in the hidden layers. As a result, we constructed and fine-tuned the eight-hidden-layer (1024-1024-1024-1024-1024-60-500-2049) DNN for *STACK*. We used the sigmoid function for all

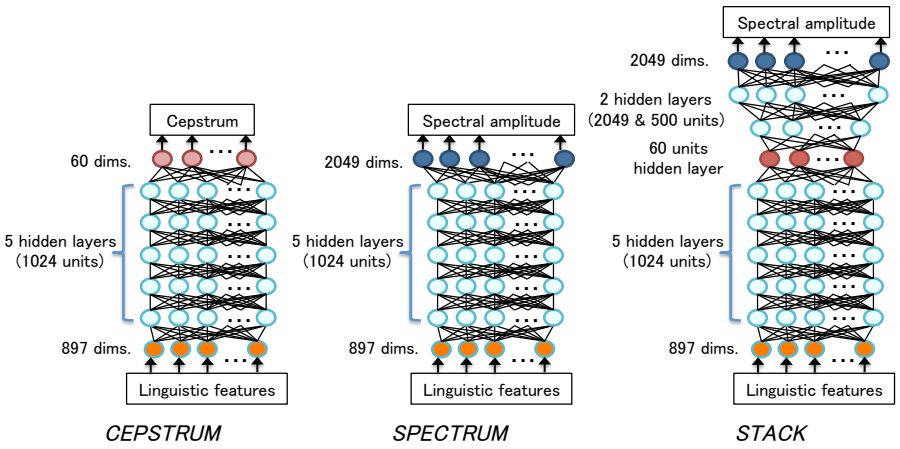


Fig. 12. Structures of constructed DNNs for each technique.

the units of hidden and output layers of all DNNs.

The same acoustic and linguistic features mentioned in the previous text-to-speech experiment were used to construct each system, although the dynamic and acceleration features were not used as acoustic features this time. This was because the auto-encoder only used static features and the two DNNs to be combined to use the same features. Note that all the techniques only synthesized spectrum features and other requisite acoustic features; that is, F0 and aperiodicity energies were synthesized from the same HMM-based synthesis system [1]. The input and output features were normalized in the same way as those in the previous text-to-speech experiment. In the proposed technique the bottleneck features were not normalized, and the normalization process was not used for hidden units in the stacked DNN. In *CEPSTRUM* synthesized cepstral vectors were converted into spectrum amplitudes to use the STRAIGHT vocoder.

Preference tests were conducted in subjective experiments. Eighteen native speaking subjects and seven non-native speaking subjects have participated in these listening tests. Fifteen and thirty sentences were randomly selected from the test set for each subject in native and non-native listening tests, respectively. The subjects were asked to listen to a pair of speech samples to answer which samples sound more natural. The experiments were carried out using headphones in a quiet room.

7.2. Experimental Results

We conducted two subjective tests to perceptually evaluate the effectiveness of the proposed technique. Figures 13 and 14 show the results of the preference tests with 95% confidence intervals of native and non-native listeners. In the first preference test, they were asked to compare the DNN that synthesized cepstrum vectors (*CEPSTRUM*) with the proposed DNN (*STACK*). In the second preference test,

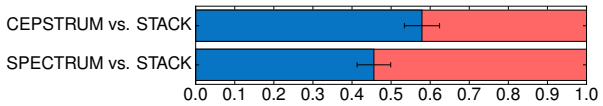


Fig. 13. Results from preference tests using text-to-speech samples (native listeners).

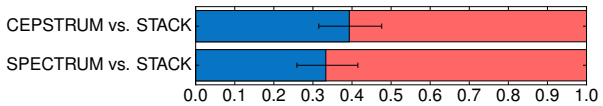


Fig. 14. Results from preference tests using text-to-speech samples (non-native listeners).

they were asked to compare the DNN without the proposed pre-training technique that synthesized spectrum amplitudes (*SPECTRUM*) with the proposed DNN (*STACK*).

From the figures, we first observe that the native listeners slightly prefer the CEPSTRUM system while the non-native listeners slightly prefer the STACK system interestingly. This will require further investigation. Then we can see that the proposed technique (STACK) produces more natural-sounding speech than the SPECTRUM system and their differences are statistically significant in both the experiments. These results indicate that the DNN that directly synthesized spectral amplitude was efficiently trained using the proposed technique and accurately synthesized spectral fine structures.

8. CONCLUSIONS

In this paper, we have proposed a technique of constructing a DNN that directly synthesizes spectral amplitudes. On the basis of the general flow for constructing the statistical parametric speech synthesis systems, part of the layers of a DNN could be efficiently pre-trained using neural networks

for data-driven non-linear feature extraction from spectral amplitudes and acoustic modeling. The experimental results shown that the spectral parameters found from the deep auto-encoder were useful in speech synthesis and the proposed technique increased the quality of synthetic speech.

In future work, we will investigate the effects of structures of a DNN-based acoustic model and a DNN-based spectrum auto-encoder more thoroughly. Time derivative features should also be interesting to investigate.

9. REFERENCES

[1] H. Zen, K. Tokuda, and A. W. Black, "Statistical parametric speech synthesis," *Speech Communication*, vol. 51, pp. 1039–1064, 2009.

[2] T. Yoshimura, K. Tokuda, T. Masuko, T. Kobayashi, and T. Kitamura, "Speaker interpolation in HMM-based speech synthesis system," *Proceedings of Eurospeech 1997*, pp. 2523–2526, 1997.

[3] T. Yoshimura, K. Tokuda, T. Masuko, T. Kobayashi, and T. Kitamura, "Simultaneous modeling of spectrum, pitch and duration in HMM-based speech synthesis," *Proceedings of Eurospeech 1999*, pp. 2347–2350, 1999.

[4] R. Tsuzuki, H. Zen, K. Tokuda, T. Kitamura, M. Bulut, and S. Narayanan, "Constructing emotional speech synthesizers with limited speech database," *Proceedings of ICSLP*, vol. 2, pp. 1185–1188, 2004.

[5] J. Yamagishi, K. Onishi, T. Masuko, and T. Kobayashi, "Acoustic modeling of speaking styles and emotional expressions in HMM-based speech synthesis," *IEICE Transactions on Information & Systems*, vol. E88-D, no. 3, pp. 502–509, 2005.

[6] H. Zen, A. Senior, and M. Schuster, "Statistical parametric speech synthesis using deep neural networks," *Proceedings of ICASSP*, pp. 7962–7966, 2013.

[7] Z.-H. Ling, L. Deng, and D. Yu, "Modeling spectral envelopes using restricted Boltzmann machines and deep belief networks for statistical parametric speech synthesis," *Audio, Speech, and Language Processing, IEEE Transactions on*, vol. 21, pp. 2129–2139, 2013.

[8] Y. Fan, Y. Qian, F. Xie, and F. K. Soong, "TTS synthesis with bidirectional LSTM based recurrent neural networks," *Proceedings of Interspeech*, pp. 1964–1968, 2014.

[9] R. Fernandez, A. Rendel, B. Ramabhadran, and R. Hoory, "Prosody contour prediction with long short-term memory, bi-directional, deep recurrent neural networks," *Proceedings of Interspeech*, pp. 2268–2272, 2014.

[10] S. Vishnubhotla, R. Fernandez, and B. Ramabhadran, "An autoencoder neural-network based low-dimensionality approach to excitation modeling for HMM-based text-to-speech," *Proceedings of ICASSP*, pp. 4614–4617, 2010.

[11] L.-H. Chen, T. Raitio, C. Valentini-Botinhao, J. Yamagishi, and Z.-H. Ling, "DNN-based stochastic postfilter for HMM-based speech synthesis," *Proceedings of Interspeech*, pp. 1954–1958, 2014.

[12] H. Kawahara, I. Masuda-Katsuse, and A. Cheveigne, "Restructuring speech representations using a pitch-adaptive time-frequency smoothing and an instantaneous-frequency-based F0 extraction: Possible role of a repetitive structure in sounds," *Speech Communication*, vol. 27, pp. 187–207, 1999.

[13] S. Hochreiter, Y. Bengio, P. Frasconi, and J. Schmidhuber, "Gradient flow in recurrent nets: the difficulty of learning long-term dependencies," *Citeseer*, 2001.

[14] G.E. Hinton, "Learning multiple layers of representation," *Trends in Cognitive Sciences*, vol. 11, pp. 428–434, 2007.

[15] G. E. Hinton and R. Salakhutdinov, "Reducing the dimensionality of data with neural networks," *Science 28*, vol. 313, no. 5786, pp. 504–507, 2006.

[16] P. Vincent, H. Larochelle, Y. Bengio, and P. Manzagol, "Extracting and composing robust features with denoising autoencoders," *ICML*, pp. 1096–1103, 2008.

[17] Chris M. Bishop, "Training with noise is equivalent to tikhonov regularization," *Neural Computing*, vol. 7, no. 1, pp. 108–116, Jan. 1995.

[18] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Parallel distributed processing: Explorations in the microstructure of cognition, vol. 1," pp. 318–362, 1986.

[19] T. N. Sainath, B. Kingsbury, and B. Ramabhadran, "Auto-encoder bottleneck features using deep belief networks," *Proceedings of ICASSP*, pp. 4153–4156, 2012.

[20] J. Gehring, Y. Miao, F. Metze, and A. Waibel, "Extracting deep bottleneck features using stacked auto-encoders," *Proceedings of ICASSP*, pp. 3377–3381, 2013.

[21] A. L. Maas, Q. V. Le, T. M. ONeil, O. Vinyals, P. Nguyen, and A. Ng Andrew, "Recurrent neural networks for noise reduction in robust ASR," *Proceedings of Interspeech*, pp. 22–25, 2012.

[22] T. Ishii, H. Komiyama, T. Shinozaki, Y. Horiuchi, and S Kuroiwa, "Reverberant speech recognition based on denoising autoencoder," *Proceedings of Interspeech*, pp. 3512–3516, 2013.

[23] X. Feng, Y. Zhang, and J. Glass, "Speech feature denoising and dereverberation via deep autoencoders for noisy reverberant speech recognition," *Proceedings of ICASSP*, pp. 1778–1782, 2014.

[24] L. Deng, M. Seltzer, D. Yu, A. Acero, A. Mohamed, and G. Hinton, "Binary coding of speech spectrograms using a deep auto-encoder," *Proceedings of Interspeech*, pp. 1692–1695, 2010.

[25] X. Lu, Y. Tsao, S. Matsuda, and C. Hori, "Speech enhancement based on deep denoising autoencoder," *Proceedings of Interspeech*, pp. 436–440, 2013.

[26] N. Kumar and A. G. Andreou, "Heteroscedastic discriminant analysis and reduced rank HMMs for improved speech recognition," *Speech Communication*, pp. 283–297, 1998.

[27] M. J. F. Gales, "Maximum likelihood multiple subspace projections for hidden Markov models," *Speech and Audio Processing, IEEE Transactions on*, vol. 10, pp. 37–47, 2002.

[28] S. J. D. Prince and J. H. Elder, "Probabilistic linear discriminant analysis for inferences about identity," *ICCV*, pp. 1–8, 2007.

[29] P. Kenny, "Bayesian speaker verification with heavy-tailed priors," *Odyssey*, p. 14, 2010.

[30] L. Lu and S. Renals, "Probabilistic linear discriminant analysis for acoustic modelling," *Signal Processing Letters, IEEE*, pp. 702–706, 2014.

[31] T. Raitio, A. Suni, L. Juvela, M. Vainio, and P. Alku, "Deep neural network based trainable voice source model for synthesis of speech with varying vocal effort," *Proceedings of Interspeech*, pp. 1969–1973, 2014.

[32] P. K. Muthukumar and Black. A., "A deep learning approach to data-driven parameterizations for statistical parametric speech synthesis," *CoRR*, vol. abs/1409.8558, 2014.

[33] I. Sutskever, J. Martens, George E. Dahl, and G. E. Hinton, "On the importance of initialization and momentum in deep learning," *ICML*, pp. 1139–1147, 2013.

[34] K. Richmond, R. Clark, and S. Fitt, "On generating combilex pronunciations via morphological analysis," *Proceedings of Interspeech*, pp. 1974–1977, 2010.